

Temporal Reasoning without Transitive Tables

Sylviane R. Schwer

LIPN UMR 7030 (Université Paris 13 et CNRS)
sylviane.schwer@lipn.univ-paris13.fr

Representing and reasoning about qualitative temporal information is an essential part of many artificial intelligence tasks. Lots of models have been proposed in the literature for representing such temporal information. All derive from a point-based or an interval-based framework. One fundamental reasoning task that arises in applications of these frameworks is given by the following scheme: given possibly indefinite and incomplete knowledge of the binary relationships between some temporal objects, find the consistent scenarii between all these objects. All these models require transitive tables — or similarly inference rules — for solving such tasks.

In [30], we have defined an alternative model, renamed in [31] S-languages – for Set-languages – to represent qualitative temporal information, based on the only two relations of *precedence* and *simultaneity*. In this paper, we show how this model enables to avoid transitive tables or inference rules to handle this kind of problem.

Keywords: Temporal reasoning, formal languages, constraints satisfaction.

1. Introduction

Representing and reasoning about qualitative temporal information is an essential part of many artificial intelligence tasks. These tasks appear in such diverse areas as natural language processing, planning, plan recognition, and diagnosis. Allen [1,2] has proposed an interval algebra framework and Vilain and Kautz [34] have proposed a point algebra framework for representing such qualitative information. All models that have been proposed afterwards in the literature derive from these two frameworks. Placing two intervals on the Timeline, regardless of their length, gives thirteen relations, known as Allen's [2] relations. Vilain [33]

provided relations for points and intervals, Kandrasina [18] provided relations for points, intervals and chains of intervals. Relations between two chains of intervals have been studied in depth by Ladkin who named them non convex intervals [20]. Al-Khatib [19] used a matricial approach. Ligozat [22] has studied relations between chains of points, named generalized intervals.

One fundamental reasoning task that arises in applications in these frameworks is given by the following scheme: given possibly indefinite and incomplete knowledge of the relationships between some temporal objects, find the consistent scenarii between all these objects. All these models have in common that the representations of temporal information are depicted as sets of binary relationships and are viewed as binary constraint networks. The reasoning is then based on transitive tables that describe the composition of any two binary relations. All these models require transitive tables - or similarly inference rules - for solving such tasks. The logical approach of the I.A. community explains this fact.

The framework of formal languages, inside which the model of S-languages has been proposed [30, 31], provides the same material both for expressing the temporal objects and the n-ary relationships between them. The reasoning is based on three natural extensions of very well-known rational operations on languages: the intersection, the shuffle and the projection. More precisely, we have shown in [31] that binary relations between two generalized intervals are in a natural correspondence with S-languages that express Delannoy paths of order 2. By the way, we provide to Henry Delannoy (1833-1915) a large domain of applications (though unexpected) of his theory of minimal paths of the queen from one corner to any other position on a chess-board [9].

The main idea for using formal languages for temporal representation and reasoning is that a word can be viewed as a line, with an arrow from

left to right (the way of reading in european languages). Hence, assigning a letter to each temporal object, as its identity, and using as many occurrences of this identity as it has points or interval bounds, it is possible to describe an atomic temporal relation between n objects on the timeline, as far as there is no simultaneity, with a word on an n -alphabet (alphabet with n letters). Simultaneity requires to be able to write several letters inside a same *box*. This is the aim of the theory of S-languages.

In this paper, we show how the S-languages framework allows to represent n -ary qualitative temporal relations and to reason without any transitive tables.

In the next part, we recall the basis of formal languages, following [3,13], and S-languages, and we examine the usual operations of the relational algebra [21] in the context of S-languages. We then provide two examples of how to reason without transitivity tables. The first one is a revisitation of the well-known unsatisfiable closed network of Allen [2]. The second one revisits the Manna-Pnuelli's problem of the allocation of a resource between several requesters [24]. This aims to show how a problem of concurrency for complex systems, written in modal temporal logic can be solved with the S-languages framework.

2. Formal languages

Let us first recall some basis on formal languages.

2.1. Basis

An alphabet X is a finite nonempty set of symbols called letters. A word (of length $k \geq 0$) over an alphabet X is a finite sequence x_1, \dots, x_k of letters in X . A word x_1, \dots, x_k is usually written $x_1 \dots x_k$. The unique word having no letter, *i.e.* of length zero, called the *empty* word, is denoted by ε . The length of a word f is denoted by $|f|$. The number of occurrences of a letter a in the word f is denoted by $|f|_a$. The set of all words (resp. of length n) on X is denoted by X^* (resp. X^n). Let us remark that $X^* = \bigcup_{n \geq 0} X^n$. The set of all words on X is written X^* . A subset of X^* is called a language. The empty set \emptyset is the least language

and X^* is the greatest language for the order of inclusion.

Let u and v be words in X^* . If $u = u_1 \dots u_r$ and $v = v_1 \dots v_s$ are words, then $u.v$ (usually written uv), called the *concatenation* of u and v , is the word $u_1 \dots u_r v_1 \dots v_s$. For instance let $X = \{x, y\}$, $u = xx$ and $v = yy$, then the concatenation is $uv = xxyy$. Let us notice that $uv \neq vu$. We also have to set $u^0 = \{\varepsilon\}$, $u^1 = u$, $u^{n+1} = uu^n$. One has $v.\varepsilon = \varepsilon.v = v$.

The concatenation can be extended to languages on X by setting $L.L' = \{uv | u \in L, v \in L'\}$. This operation endows 2^{X^*} with a structure of non-commutative monoid. We also have $L^0 = \{\varepsilon\}$, $L^1 = L$, $L^{n+1} = LL^n$, $L^* = \bigcup_{n \geq 0} L^n$.

$u^* = \bigcup_{n \geq 0} u^n$. Even if u^* is a set, it can be worked with like an element, so that we will take this alternative and use u^* as a word or S-word.

The shuffle is a very useful operator which is used in concurrency applications. The shuffle operator describes all possibilities of doing two concurrent sequences of actions in a sequential manner. Therefore, this is not a binary combination of X^* because, from two words, it provides a set of words, that is a language. Its definition is the following: Let u and v be two words written on an alphabet X^* . The shuffle of u and v is the language $u \mathbb{W} v = \{\alpha_1 \beta_1 \dots \alpha_k \beta_k \in X^* | \alpha_1, \beta_k \in X^*, \alpha_2, \dots, \alpha_k, \beta_1, \dots, \beta_{k-1} \in X^+, u = \alpha_1 \dots \alpha_k, v = \beta_1 \dots \beta_k\}$. For instance let $X = \{x, y\}$, then $xx \mathbb{W} yy = \{xxyy, xyxy, yxyx, xyxy, yxyx, yxyx\}$.

The concatenation uv means an order between u and v , this is a word of the language $u \mathbb{W} v$. One has $\varepsilon \mathbb{W} v = v \mathbb{W} \varepsilon = v$ for any word v of X^* . The shuffle can be naturally extended to languages on X by setting $L \mathbb{W} L' = \bigcup_{u \in L, v \in L'} u \mathbb{W} v$. The shuffle endows 2^{X^*} with a structure of commutative monoid.

Words are read from left to right, so that the reading induces a natural arrow of Time. Any occurrence of a letter can be viewed as an instant numbered by its position inside the word. Traces languages or paths expressions [8] are used for such a purpose: planning the order of execution of events. But with these languages, it is not possible to differentiate two occurrences that are concurrent (*i.e.* one may be *before*, *at the same time* or *after* the other) from those that must occur at the same *time*: these two events are said to commute. It is presupposed that the granularity of the time measurement is fine enough to avoid the case *at the same time*.

2.2. S-alphabet, S-words, S-languages

In order to model explicitly concurrency with words, various tools have been proposed such as event structures or equivalence relations on words i.e. traces. In those theories, it is not possible to model only synchronization. One is able to say that two events can be done at the same time but it is not possible to express that they have to be done at the same time. This is due to the fact that concurrency is modelled inside a deeply sequential framework, hence, synchronization is simulated with commutativity. But one has to handle with *instant*, in the sense of Russell [29]. This is why we introduce the concept of S-alphabet which is a powerset of a usual alphabet.

2.2.1. Basic definitions

Let us set

Definition 2.1 *If X is an alphabet, an S-alphabet over X is a non-empty subset of $2^X - \emptyset$. An element of an S-alphabet is an S-letter. A word on an S-alphabet is an S-word. A set of words on an S-alphabet is an S-language.*

S-letters are written either horizontally or vertically: $\{a, b\} = \{a, b, \begin{smallmatrix} a \\ b \end{smallmatrix}\}$. For S-letters with only two letters, we also write $\begin{pmatrix} a \\ b \end{pmatrix}$ instead of $\begin{smallmatrix} a \\ b \end{smallmatrix}$.

Examples of S-alphabets over X are:

1. the *natural* one $\dot{X} = \{\{a\} \mid a \in X\}$ that is identified with X .
2. the *full* S-alphabet over X , i.e. $\hat{X} = 2^X - \emptyset$.
3. S-alphabets obtained from others S-alphabets with the following construction:

For an S-alphabet Y over X , define $\widehat{Y} = \{A \mid \exists A_1, \dots, A_k \in Y : A = \bigcup_{i=1}^k A_i\}$. \widehat{Y} is also an S-alphabet over X .

Note that, for all S-alphabets Y and Z over X , we have $\widehat{\widehat{Y}} = \widehat{Y}$ and $\widehat{Y \cup Z} = \widehat{Y} \cup \widehat{Z}$. A S-word on a full S-alphabet over X will be simply designed by an S-word on X .

In this work, we use the full S-alphabet $\hat{X} = 2^X - \emptyset$. Identifying any singleton with its letter, we write $X \subset \hat{X}$ and $X^* \subset \hat{X}^*$.

In order to link S-words on X with letters of X , we set

Definition 2.2 *Let $X = \{x_1, \dots, x_n\}$ be an n -alphabet and $f \in \hat{X}^*$. We note $\|f\|_x$ for $x \in X$ the number of occurrences of x appearing inside the S-letters of f , and $\|f\|$ the integer $\sum_{1 \leq i \leq n} \|f\|_{x_i}$. The Parikh vector of f , denoted \vec{f} , is the n -tuple $(\|f\|_{x_1}, \dots, \|f\|_{x_n})$.*

Example 1 :

$$f = \begin{pmatrix} a \\ b \end{pmatrix} cba \begin{pmatrix} a \\ c \end{pmatrix} c \begin{pmatrix} a \\ b \end{pmatrix} a \begin{pmatrix} a \\ b \\ c \end{pmatrix} aaaa \text{ is an S-}$$

word such that $\vec{f} = (10, 4, 4)$.

2.2.2. Concatenation and shuffle

The concatenation of two S-words or two S-languages are defined exactly in the same way as in formal languages. The S-shuffle has to be generalized in the following way:

Definition 2.3 *Let X and Y be two disjoint alphabets, $f \in \hat{X}^*$, $g \in \hat{Y}^*$. The S-shuffle of f and g is the language $[f||g] = \{h_1 \dots h_r \mid h_i \in \widehat{X \cup Y}, \text{ with } \max(|f|, |g|) \leq r \leq |f| + |g| \text{ and such that there are decompositions of } f \text{ and } g: f = f_1 \dots f_k, g = g_1 \dots g_k, \text{ satisfying, (i) } \forall i \in [r], |f_i|, |g_i| \leq 1, \text{ (ii) } 1 \leq |f_i| + |g_i|, \text{ and (iii) } h_i = f_i \cup g_i\}$.*

For instance $[aa||bb] = \{aabb, a \begin{pmatrix} a \\ b \end{pmatrix} b, abab, \begin{pmatrix} a \\ b \end{pmatrix} ab, ab \begin{pmatrix} a \\ b \end{pmatrix}, \begin{pmatrix} a \\ b \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix}, baab, ba \begin{pmatrix} a \\ b \end{pmatrix}, abba, \begin{pmatrix} a \\ b \end{pmatrix} ba, baba, b \begin{pmatrix} a \\ b \end{pmatrix} a, bbaa\} = \{f \in \widehat{\{a, b\}}^* \mid \vec{f} = (2, 2)\}$.

The S-shuffle of two S-languages L and L' written on two disjoint alphabets is the language $[L||L'] = \cup_{f \in L, f' \in L'} [f||f']$.

The S-shuffle is, like the shuffle, an associative and commutative operations, which allows to note $[u_1||\dots||u_n]$ for the S-shuffle of n S-words or S-languages.

In the case where all S-words of a language \mathcal{L} share the same Parikh vector, we note $\vec{\mathcal{L}}$ this common Parikh vector. In particular, on the n -alphabet $X = \{x_1, \dots, x_n\}$, the language,

$$\mathcal{L}(p_1, \dots, p_n) = \{f \in \hat{X}^* \mid \vec{f} = (p_1, \dots, p_n)\}$$

that we call (p_1, \dots, p_n) -Delannoy Language – on X –, are of a particular interest for temporal qualitative reasoning, as we will show it in the next section. Let us just recall [31] that the cardinality $D(p_1, \dots, p_n)$ of a (p_1, \dots, p_n) -Delannoy Language is given by the following functional equation:

$$D(p_1, \dots, p_n) = \sum_{\text{Pred}(p_1, \dots, p_n)} D_n(\tilde{p}_1, \dots, \tilde{p}_n)$$

where for $p > 0$, $\tilde{p} = \{p, p-1\}$; $\tilde{0} = \{0\}$ and $\text{Pred}((p_1, \dots, p_n)) = \{(\tilde{p}_1, \dots, \tilde{p}_n)\} - \{(p_1, \dots, p_n)\}$.

In particular $D(p, q) = D(p, q-1) + D(p-1, q-1) + D(p-1, q)$ with the initial steps $D(0, 0) = D(0, 1) = D(1, 0) = 1$.

Like Pascal's table for computing binomial numbers, there is a Delannoy table for computing Delannoy numbers, given in Table 1.

$p \backslash q$	0	1	2	3	4	5	6	7	8
0	1	1	1	1	1	1	1	1	1
1	1	3	5	7	9	11	13	15	17
2	1	5	13	25	41	61	85	113	145
3	1	7	25	63	129	231	377	575	833
4	1	9	41	129	321	681	1289	2241	3649
5	1	11	61	231	681	1683	3653	7183	13073
6	1	13	85	377	1289	3653	8989	19825	40081
7	1	15	113	575	2241	7183	19825	48639	108545
8	1	17	145	833	3649	13073	40081	108545	265729
9	1	19	181	1159	5641	22363	75517	224143	598417

Table 1
Delannoy table

$D(p, q)$ are the well-known Delannoy numbers [9,35,32] that enumerate Delannoy paths in a (p, q) -rectangular chessboard. A Delannoy path is given as a path that can be drawn on a rectangular grid, starting from the southwest corner, going to the northeast corner, using only three kinds of elementary steps: *north*, *east*, and *north-east*. Hence they are minimal paths with diagonal steps. The natural correspondence between (p, q) -Delannoy paths and $\mathcal{L}(p, q)$ on the alphabet $\{a, b\}$ is: the S-letter a corresponds to a *north*-step, the S-letter b to a *east*-step and the S-letter $\binom{a}{b}$ to *north-east*-step.

2.2.3. Projection

We extend the well-known notion of projection in formal languages theory to S-languages. The aim is to be able to erase all occurrences of some letters in an S-word and having as results a new S-word. The problem is how to handle an S-letter with all its letters erased. For that purpose we set: let X be an alphabet and $f \in \hat{X}^*$, $X_f = \{x \in X \mid \|f\|_x \neq 0\}$.

Definition 2.4 Let X be an alphabet and $Y \subseteq X$. The S -projection from \hat{X}^* to \hat{Y}^* is a monoid morphism π_Y^X defined by the image of the S -letters: for $s \in X$, its image is $\pi_Y^X(s) = s \cap Y$ if this intersection is not empty, ε if not.

The projection on Y of an S-word f is denoted $f|_Y$ instead of $\pi_Y^X(f)$.

Example 2 (Example 1 continued)

$$\text{Let } f = \left\{ \begin{smallmatrix} a \\ b \end{smallmatrix} \right\} cba \left\{ \begin{smallmatrix} a \\ c \end{smallmatrix} \right\} c \left\{ \begin{smallmatrix} a \\ b \end{smallmatrix} \right\} a \left\{ \begin{smallmatrix} a \\ b \\ c \end{smallmatrix} \right\} aaaa,$$

- $f|_{\{a\}} = aaaaaaaaaa$,
- $f|_{\{b\}} = bbbb$,
- $f|_{\{c\}} = cccc$,
- $f|_{\{a,b\}} = \left\{ \begin{smallmatrix} a \\ b \end{smallmatrix} \right\} baa \left\{ \begin{smallmatrix} a \\ b \end{smallmatrix} \right\} a \left\{ \begin{smallmatrix} a \\ b \end{smallmatrix} \right\} aaaa$,
- $f|_{\{b,c\}} = bcbccb \left\{ \begin{smallmatrix} b \\ c \end{smallmatrix} \right\}$,
- $f|_{\{a,c\}} = aca \left\{ \begin{smallmatrix} a \\ c \end{smallmatrix} \right\} caa \left\{ \begin{smallmatrix} a \\ c \end{smallmatrix} \right\} aaaa$,
- $f|_{\{a,b,c\}} = f$.

3. Qualitative Temporal Objects and Relations in the binary algebra and their transitivity tables

We examine qualitative temporal objects and relations inside the framework of relational algebra, as Ladkin and Maddux initiated it [21]. We recall the usual qualitative temporal binary algebra: the point algebra, the interval algebra [2], the point-interval algebra [33,34], chains algebra.

In this paper, we use the term *situation* for the description of a unique temporal relation (complete information) between objects, which is sometimes called an atomic relation.

Let us recall the principia of transitivity table. Given a particular theory Σ supporting a set of mutually exhaustive and pairwise disjoint dyadic situations, three individuals, a , b and c and a pair of dyadic relations R_1 and R_2 selected from Σ such that $R_1(a, b)$ and $R_2(b, c)$, the transitive closure $R_3(a, c)$ represents a disjunction of all the possible dyadic situations holding between a and c in Σ . Each $R_3(a, c)$ result can be represented as one entry of a matrix for each $R_1(a, b)$ and $R_2(b, c)$ ordered pair. If there are n dyadic situations supported by Σ , then there will be $n \times n$ entries in the matrix. This matrix is a transitivity table. Transitive tables for binary situations have been written

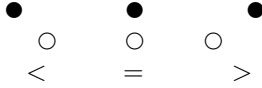
for convex intervals (Allen), for points, for points and convex intervals: knowing an atomic relation between objects A and B and an atomic relation between objects B and C , derive the possible - or other said not prohibited - relations between objects A and C .

Cohn *et al.* [28] have studied transitivity tables for reasoning in both time and space in a more general context. They noted the difficulty to build such secure transitivity tables.

3.1. The Point Algebra.

The three situations are the three basic temporal relations: before $<$, equals $=$ and after $>$ as shown in Figure 1. The set of point qualitative

Fig. 1. situations of the black point with respect to the white point.



temporal binary relations is the set : $\{<, >, =, \leq, \geq, \neq, \perp, \top\}$, where \perp is the empty relation (no feasible relation) and \top the universal relation (any relation is feasible). The transitive table is given in Figure 2

\circ	$<$	$=$	$>$
$<$	$<$	$<$	\top
$=$	$<$	$=$	$>$
$>$	\top	$>$	$>$

Fig. 2. point transitivity table

For instance, if $A < B$ and $B > C$ then $A \top C$.

3.2. The Interval Algebra.

In Figure 3, we recall the thirteen situations between two intervals studied by Allen [1].

The transitivity table is given in Table 4 where¹: $\square = \top$, that is there is no constraint, every situa-

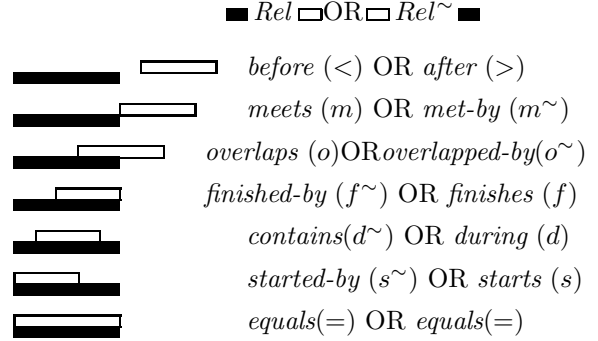


Fig. 3. the set of 13 situations between two intervals on line

tion is allowed,

$\diamond = \top - \{<, m, m~, >\}$, which means that the two intervals intersects on an interval, this is exactly what Kamp named the overlapping² relation \circ on two processes [17] and Freksa the contemporary relation [14],

$\Gamma = \{<, m, o, s, d\}$, that is the relation *begin before*,

$\Lambda = \{<, m, o, d~, f~\}$, that is the relation *end after*

$\alpha = \{<, m, o\}$, $\delta = \{o, s, d\}$, $\rho = \{o~, d, f\}$,

$\hat{s} = \{s, =, s~\}$, $\hat{f} = \{f, =, f~\}$

and using the following property:

$\forall A \subseteq \top, x \in A \text{ iff } x~ \in A~$.

\circ	$=$	$<$	$>$	d	$d~$	o	$o~$	m	$m~$	s	$s~$	f	$f~$
$=$	$=$	$<$	$>$	d	$d~$	o	$o~$	m	$m~$	s	$s~$	f	$f~$
$<$	$<$	$<$	\square	Γ	$<$	$<$	Γ	$<$	Γ	$<$	$<$	Γ	$<$
$>$	$>$	\square	$>$	Λ	$>$	Λ	$>$	Λ	$>$	Λ	$>$	$>$	$>$
d	d	$<$	$>$	d	\square	Γ	Λ	$<$	$>$	d	Λ	d	Γ
$d~$	$d~$	Λ	Γ	\diamond	$d~$	ρ	δ	ρ	δ	ρ	$d~$	δ	$d~$
o	o	$<$	Γ	δ	Λ	α	\diamond	$<$	δ	o	ρ	δ	α
$o~$	$o~$	Λ	$>$	ρ	Γ	\diamond	α	$>$	ρ	$>$	ρ	α	δ
m	m	$<$	Γ	δ	$<$	$<$	δ	$<$	\hat{f}	m	m	δ	$<$
$m~$	$m~$	Λ	$>$	ρ	$>$	ρ	$>$	\hat{s}	$>$	ρ	$>$	$m~$	$m~$
s	s	$<$	$>$	d	Λ	α	ρ	$<$	$m~$	s	\hat{s}	d	α
$s~$	$s~$	Λ	$>$	ρ	$d~$	ρ	$o~$	ρ	$m~$	\hat{s}	$s~$	$o~$	$d~$
f	f	$<$	$>$	d	Γ	δ	α	m	$>$	d	α	f	f
$f~$	$f~$	$<$	Γ	δ	$d~$	o	δ	$m~$	δ	o	$d~$	f	$f~$

Fig. 4. interval transitivity table

¹The notation is taken, whenever possible, from Delannoy paths draw as kind of greek letters on the chess, with the following convention: Upper case for 5-subsets, lower case for 3-subsets

²In french, couverture

3.3. The Point-Interval Algebra

In order to take into account both instantaneous and durative processes, Vilain provided a model with points and intervals [33]. Figure 5 shows the five situations between a point and an interval.



Fig. 5. The 5 situations between a point and an interval (each point designs a situation)

Besides the two preceding transitivity tables, are needed six more transitivity tables:

- (i) points/intervals-intervals/points,
- (ii) points/intervals-intervals/intervals,
- (iii) points/points-points/intervals,
- (iv) intervals/intervals-intervals/points,
- (v) intervals/points-points/intervals,
- (vi) intervals/points-points/points.

3.4. Chains Algebras

The T-model of Kandrashina [18] has three qualitative basic notions: the point, the interval and the sequence of intervals. Situations between two sequences of intervals are derived from situations between intervals. Some frequent situations are shown like the one in Figure 6:



Fig. 6. S_1 alternates S_2

These objects has been revisited and studied for their own by Ladkin [20] under the name of non-convex intervals. Ligozat [22] generalized to sequences of points and/or intervals under the name of generalized intervals.

There are 3 situations between two points, 5 between a point and an interval, 13 situations between two intervals, 8989 situations between two sequences of three intervals or two sequences of 6 points. Ladkin [20, Theorem1], proved the number of situations between two chains of intervals is at least exponential in the number of intervals. The exact number of situations between a sequence of p points and a sequence of q points has been pro-

vided by [6, p. 83] without doing the connection with Delannoy numbers.

Freksa studied transitivity tables with respect to convex set of intervals [14], Randel & al. [28] have studied transitivity tables for reasoning in both time and space in a more general context, both in order to below the complexity rate of the computations. That was also the aim of Vilain *et al.* who have studied the fragment of the interval algebra, that can be written without disjunction inside the point algebra, based on the fact that relations between intervals can be translated in terms of their bounds, inside the point algebra. An interval A is a couple of its bounds (a, \bar{a}) viewed as points they can contain or not, with the constraint $a < \bar{a}$. Situations between intervals are represented in terms of the situations of their bounds:

- A is *before* B iff $a < \bar{a} < b < \bar{b}$
- A *meets* B iff $a < \bar{a} = b < \bar{b}$
- A *overlaps* B iff $a < b < \bar{a} < \bar{b}$
- A *starts* iff B $a = b < \bar{a} < \bar{b}$
- A *during* iff $b < a < \bar{a} < \bar{b}$
- A *finishes* iff $b < a < \bar{a} = \bar{b}$
- A equals B iff $a = b < \bar{a} = \bar{b}$.

4. Qualitative Temporal Objects and Relations in the S-languages framework

4.1. Temporal Objects

All temporal items previously reviewed are based on points or maximal convex interval, that is isolated points or pairing points. The idea is to assign an identity to each temporal objects. The set of these identities is the alphabet on which the S-languages will be written. A temporal object with identity a and p bounds and/or isolated points is depicted by the (S-)word a^p . To distinguish between points and bounds, it is possible to mark the right bound of an interval. If one non-marked letter follows a non-marked letter, then the first one depicts a point. For instance the S-word $aa\bar{a}aaaa\bar{a}\bar{a}$ depicts the sequence : point, interval, point, point, interval, interval.



Fig. 7. A relation between 3 chains of intervals

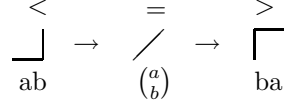


Fig. 8. Point lattice

4.2. Temporal Relations

A relation between n temporal items, using alphabet $X = \{x_1, \dots, x_n\}$, is an S-word on $\hat{X}^* = 2^X - \emptyset$ that describes exactly the situation of the points on the timeline, described by the relation. For instance

Example 3 (Examples 1 and 2 continued) Let A, B, C three temporal items as depicted in figure 7.

On the alphabet $X = \{a, b, c\}$, item A is written $aaaaaaaaa$, item B $bbbb$ and item C $cccc$. The situation between them is given by the S-word f of Example 1, that is

$$f = \left\{ \begin{matrix} a \\ b \end{matrix} \right\} cba \left\{ \begin{matrix} a \\ c \end{matrix} \right\} c \left\{ \begin{matrix} a \\ b \end{matrix} \right\} a \left\{ \begin{matrix} a \\ b \\ c \end{matrix} \right\} aaaa. \text{ Hence, in}$$

Example 2, we have computed:

- $f_{\{a\}}$, which is item A ,
- $f_{\{b\}}$, which is item B ,
- $f_{\{c\}}$, which is item C ,
- $f_{\{a,b\}}$, which is the relation between A and B ,
- $f_{\{b,c\}}$, which is the relation between B and C ,
- $f_{\{a,c\}}$, which is the relation between A and C ,
- $f_{\{a,b,c\}}$, which is the relation between A, B and C .

The following theorem [31] is the most important for our purpose:

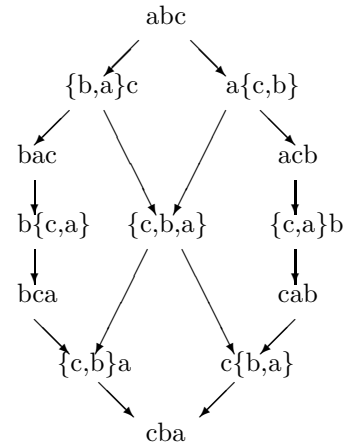
Theorem 4.1 For any integer $n \geq 1$, let T_1, \dots, T_n be n temporal items, $X_n = \{x_1, \dots, x_n\}$ be an alphabet and x^{p_1}, \dots, x^{p_n} - writing x^n the word $\underbrace{x \dots x}_{n \text{ times}}$ - their temporal words on X . Let us denote by $\Pi(p_1, \dots, p_n)$ the set of all n -ary situations among T_1, \dots, T_n , $\mathcal{L}(p_1, \dots, p_n)$ is its corresponding language.

In dimension 2, it is obvious to see that there is a natural correspondence between $\mathcal{L}(p, q)$ and Delannoy paths in a (p, q) -rectangular chessboard. The correspondence between interval situations, $\mathcal{L}(p, q)$ and $(2, 2)$ -Delannoy paths is shown in Figure 9, inside the Nökel Lattice [25].

The arrow means, for S-words, the Thue rewriting rules [3] $ab \rightarrow \begin{pmatrix} a \\ b \end{pmatrix} \rightarrow ba$, which is exactly the

Point lattice as we can see it in Figure 8. Autebert *et al.* have proved [4] that the S-language $\mathcal{L}(p, q)$ on the alphabet $\{a, b\}$ (that is any set of situations between a sequence of p points and a sequence of q points or Ligozat's $\Pi(p, q)$ set [22]) can be generated from the single S-word $a^p b^q$ and these Thue rewriting rules. They also rigorously proved that these rules make the (p, q) Parikh vector S-language to be a distributive lattice. They also characterize the subset of *union-irreducible* S-words, which is the lattice of ideals of the language : $\{a^{p-1}b^{q-1} \mid p > 0, q > 0\} \cup \{a^{p-k}b^l a^k b^{q-l} \mid 0 < l \leq q, 0 < k \leq p\}$. Its cardinality is $2pq$.

Autebert and Schwer [5] generalized the results to the n -ary case, proving that $\mathcal{L}(p_1, \dots, p_n)$, with the following Thue rewriting rule is also a lattice, but not distributive because not modular, as soon as $n \geq 3$ like Figure 10 shows it. Given an arbitrary order over the letters of X by $a_1 < a_2 < \dots < a_n$, this induces over the S-letters a partial order $P < Q \iff [\forall x \in P, \forall y \in Q : x < y]$. Then the Thue system denoted \longrightarrow on \hat{X}^* , by the following: $\forall P, Q, R \in \hat{X}$ such that $P < Q$ and $R = P \cup Q$, set $PQ \longrightarrow R$ and $R \longrightarrow QP$.

Fig. 10. $\mathcal{L}(1, 1, 1)$ is a non modular lattice.

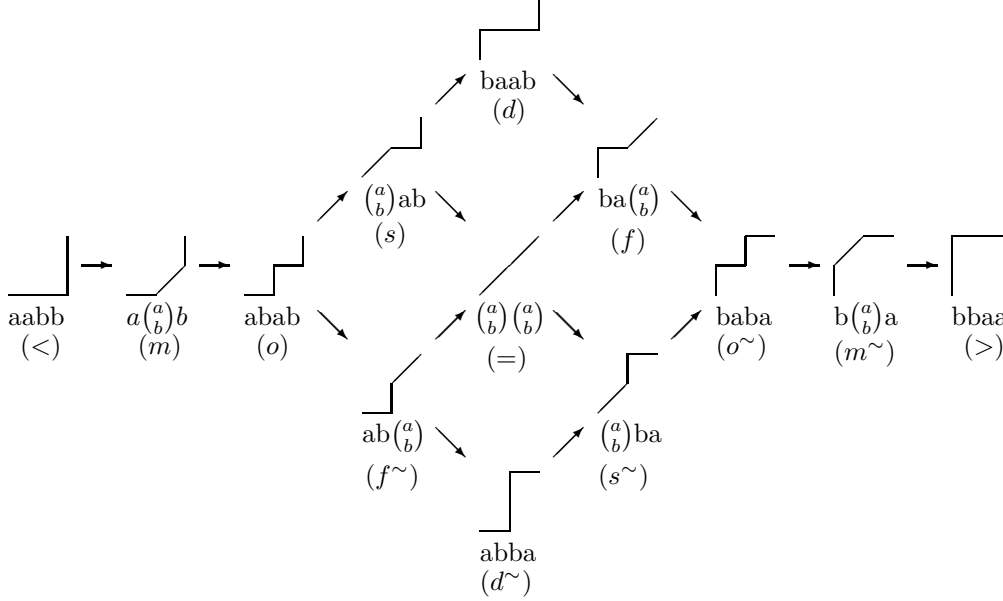


Fig. 9. The Nökel Lattice for the interval Algebra

4.3. Operations on temporal relations

In a relational algebra, relations are basic objects on which operators operate. Apart of sets operators like union, intersection and complementation, we have yet seen two operations : the composition \circ and the inverse \sim . The inverse operation exchanges the role of the objects: $aSb \iff b(S)\sim a$. There is an other unary operation, closed to the Time arrow: the *symmetry* function that inverses the arrow of Time. The symmetrical of aSb is $aS\sim b$. In the framework of S-languages, the *transposition* is the identity function; the *symmetry* function is the mirror one that is the reading from right to left.

The third operation, the *composition*, is the fundamental operation for the reasoning. We show now how the S-language framework avoids such a material.

These operations have their correspondents inside the S-languages framework, but we prefer to simulate them with the two new operators from S-words to S-languages that we now introduce. The first one is the inverse of the projection, named integration, it is an unary operator; the second one is the main operator, it is closed to the composition of relations. It aims to answer the following question: having three worlds X, Y, Z – with possible intersections –, and having information f in

world X and information g in world Y, what possible – i.e. not forbidden – information can be deduced from them in world Z? This operator is the one which allows to avoid transitive table.

Definition 4.2 For any alphabet Z and any word $f \in \widehat{X}^*$,

- The free integration of the S-word f on the alphabet Z , denoted $\int_Z f$, is the S-language

$$\int_Z f = [\pi_{X_f \cup Z}^{X_f \cup Z}]^{-1}(f) = \{g \in \widehat{Z \cup X_f}^* \mid g|_{X_f} = f\}$$

- For any distinct letters t_1, \dots, t_n of Z , and $\nu = (t_1^{p_1}, \dots, t_n^{p_n})$, the bounded to ν integration of the S-word f on the alphabet Z , denoted $\int_Z^\nu f$, is the S-language

$$\int_Z^\nu f = \left(\int_Z f \right) \cap \left(\int_{Z \cup X_f} \mathcal{L}(p_1, \dots, p_n) \right) = \{g \in \widehat{Z \cup X_f}^* \mid g|_{X_f} = f, \forall i : \|g\|_{t_i} = p_i\}$$

These definitions are extended to languages in the following natural way:

The free integration of the S-language L on the alphabet Z , denoted $\int_Z L$, is the S-language

$$\int_Z L = \bigcup_{f \in L} \int_Z f$$

The bounded to ν integration of the S-language L on the alphabet Z , denoted $\int_Z^\nu L$, is the S-language

$$\int_Z^\nu L = \bigcup_{f \in L} \int_Z^\nu f$$

For instance - cf. end of Section 2 - let $Z = \{a, b, c\}$ and $\nu = (a^{10}, b^4, c^4)$. $\int_Z^\nu a^{10} = \int_Z^\nu b^4 = \int_Z^\nu c^4 = \mathcal{L}_Z(10, 4, 4)$.

$\int_Z^\nu f_{\{a,b\}} = [f_{\{a,b\}} || c^4]$ and $\int_Z^\nu f_{\{b,c\}} = [f_{\{b,c\}} || a^{10}]$.

We then have

$$\int_Z^\nu f_{\{a,b\}} \cap \int_Z^\nu f_{\{b,c\}} = \left\{ \begin{matrix} a \\ b \end{matrix} \right\} cb([aa || cc]). \left\{ \begin{matrix} a \\ b \end{matrix} \right\} a \left\{ \begin{matrix} a \\ b \\ c \end{matrix} \right\} aaaa$$

which contains the word f .

We never compute the integration. It is just an artifact in order to have every constraints written on the same alphabet. The operation which costs the most is the intersection. In fact, we do not do it. We operate a kind of join, which consists in (i) computing the set of letters in common under the two integrals, (ii) verifying if all occurrences of these letters are ordered in the same manner under the two integrals (iii) shuffle the two subwords which are between two such following occurrences.

(i) and (ii) causes no problem. If the common letters are isolated (that is, not inside a shuffle part), the complexity of (iii) is linear but in the worse case, it can be exponential. We are studying convex part of lattices and heuristics in order to improve the complexity of the computation, in the spirit of [11].

5. Reasoning inside the S-language framework

It is usual in temporal applications that information arrives from many various sources or a same source can complete the knowledge about a same set of intervals. The usual way to deal with that, when no weight of credibility or plausibility is given, is to intersect all the information. The knowledge among some set of intervals interferes with some other sets of intervals by transitivity: if you know that Marie leaved before your arrival, and you are waiting for Ivan who attempts to see Marie, you can tell him that he has missed her.

Vilain and Kautz [34] argued that there are two kinds of problems:

Problem number 1 Let $R_1(A, C)$ and $R_2(A, C)$ be two sets of constraints between intervals A and C , what is the resulting set of constraints for A and C ?

Problem number 2 Let A, B, C be three intervals and $R(A, B)$ and $R(B, C)$ the sets of constraints respectively between A and B and between B and C . What is the deduced set of constraints between A and C ?

The first problem requires an **and** logical operator or an **intersection** set operator. The second problem requires a transitivity operator based on tables.

In our framework, the answers to these two problems are described exactly in the same manner, the difference being just a matter of integration alphabet. Let $\mathcal{R}_1(a, b)$ [resp. $\mathcal{R}_2(b, c)$, $\mathcal{R}(a, c)$] be the language associated to $R_1(a, b)$ [resp. $R_2(b, c)$, $\mathcal{R}(a, c)$], the first answer is

$$\mathcal{R}(a, c) = \pi_Z^X \left(\int_X^{(2,2,2)} \mathcal{R}_1(a, b) \cap \int_X^{(2,2,2)} \mathcal{R}_2(b, c) \right)$$

and the second answer is

$$\mathcal{R}(a, c) = \pi_Z^X \left(\int_X^{(2,2,2)} \mathcal{R}_1(a, b) \cap \int_X^{(2,2,2)} \mathcal{R}_2(b, c) \right)$$

with in both cases $Z = \{a, c\}$. More generally, our main result, set in terms of intersection and integration, is:

Theorem 5.1 *Let $I = \{I_1, \dots, I_n\}$ be a set of n temporal items, $X = \{x_1, \dots, x_n\}$ be the corresponding alphabet and $\nu = (x_1^{p_1}, \dots, x_n^{p_n})$ their Parikh vector. Let J_1, \dots, J_k be k non empty subsets of I and Y_1, \dots, Y_k their corresponding alphabets and ν_{Y_i} their Parikh vectors. For $1 \leq i \leq k$, let $\{\mathcal{L}_{i_1}, \dots, \mathcal{L}_{i_{s_i}}\} \subseteq \mathcal{L}(\nu_{Y_i})$ be a set of languages describing s_i -ary temporal qualitative constraints among J_i .*

- *The all solution problem for I is given by the language*

$$\bigcap_{\substack{1 \leq i \leq k \\ 1 \leq j \leq s_i}} \int_X^{\nu_X} \mathcal{L}_{i_j}$$

- *The temporal satisfaction problem for I is satisfied if and only if*

$$\bigcap_{\substack{1 \leq i \leq k \\ 1 \leq j \leq s_i}} \int_X^{\nu_X} \mathcal{L}_{I_{ij}} \neq \emptyset$$

In order to have a look on what kind of language is computed, let us revisit the unsatisfiable closed network of [2]. There are four intervals A, B, C,

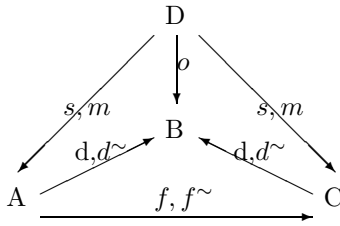


Fig. 11. Allen's instance of an inconsistent labeling.

D. We then take the alphabet $X = \{a, b, c, d\}$ and $\mathcal{L}(2, 2, 2, 2)$ on X . The data are :

$$L_1 = \int_X^\nu \left\{ \begin{Bmatrix} a \\ d \end{Bmatrix} da, d \begin{Bmatrix} a \\ d \end{Bmatrix} a \right\}$$

$$L_2 = \int_X^\nu \left\{ \begin{Bmatrix} c \\ d \end{Bmatrix} dc, d \begin{Bmatrix} c \\ d \end{Bmatrix} c \right\}$$

$$L_3 = \int_X^\nu dbdb$$

$$L_4 = \int_X^\nu \left\{ ca \begin{Bmatrix} a \\ c \end{Bmatrix}, ac \begin{Bmatrix} a \\ c \end{Bmatrix} \right\}$$

$$L_5 = \int_X^\nu \{bccb, cbbc\}$$

$$L_6 = \int_X^\nu \{adda, daad\}$$

The solution is

$$L = L_1 \cap L_2 \cap L_3 \cap L_4 \cap L_5 \cap L_6$$

that we now compute.

$$L_1 \cap L_2 =$$

$$\int_X^\nu \left\{ \begin{Bmatrix} a \\ c \\ d \end{Bmatrix} d, d \begin{Bmatrix} a \\ c \\ d \end{Bmatrix}, \begin{Bmatrix} a \\ d \end{Bmatrix} \begin{Bmatrix} c \\ d \end{Bmatrix}, \begin{Bmatrix} c \\ d \end{Bmatrix} \begin{Bmatrix} a \\ d \end{Bmatrix} \right\} [a||c]$$

or equivalently, due to the lack of occurrences of b , $L_1 \cap L_2 =$

$$\left[\left\{ \begin{Bmatrix} a \\ c \\ d \end{Bmatrix} d, d \begin{Bmatrix} a \\ c \\ d \end{Bmatrix}, \begin{Bmatrix} a \\ d \end{Bmatrix} \begin{Bmatrix} c \\ d \end{Bmatrix}, \begin{Bmatrix} c \\ d \end{Bmatrix} \begin{Bmatrix} a \\ d \end{Bmatrix} \right\} \right] [a||c||bb]$$

This language contains 164 words.

$$L_1 \cap L_2 \cap L_3 =$$

$$\left\{ \begin{Bmatrix} a \\ c \\ d \end{Bmatrix} bd, db \begin{Bmatrix} a \\ c \\ d \end{Bmatrix}, \begin{Bmatrix} a \\ d \end{Bmatrix} b \begin{Bmatrix} c \\ d \end{Bmatrix}, \begin{Bmatrix} c \\ d \end{Bmatrix} b \begin{Bmatrix} a \\ d \end{Bmatrix} \right\}.$$

$$[a||b||c]$$

This language contains 52 words.

$$L_1 \cap L_2 \cap L_3 \cap L_4 =$$

$$\left\{ \begin{Bmatrix} a \\ d \end{Bmatrix} b \begin{Bmatrix} c \\ d \end{Bmatrix}, \begin{Bmatrix} c \\ d \end{Bmatrix} b \begin{Bmatrix} a \\ d \end{Bmatrix} \right\} \cdot \left[\begin{Bmatrix} a \\ c \end{Bmatrix} ||b \right]$$

This language contains 6 words.

$$L_1 \cap L_2 \cap L_3 \cap L_4 \cap L_5 =$$

$$\left\{ \begin{Bmatrix} a \\ d \end{Bmatrix} b \begin{Bmatrix} c \\ d \end{Bmatrix} \begin{Bmatrix} a \\ c \end{Bmatrix}, \begin{Bmatrix} c \\ d \end{Bmatrix} b \begin{Bmatrix} a \\ d \end{Bmatrix} b \begin{Bmatrix} a \\ c \end{Bmatrix} \right\}$$

This language has 2 words.

$$L_1 \cap L_2 \cap L_3 \cap L_4 \cap L_5 \cap L_6 = \emptyset$$

This language is empty: the problem is unsatisfiable.

6. Temporal reasoning about concurrent systems

Most of the temporal properties of programs have been studied inside either the framework of temporal logics or modal logic. Temporal reasoning about concurrent systems can be partitioned in a natural way into two classes related to the modality used : necessity, symbolized by \Box or possibility, symbolized by \Diamond . Translated in a temporal framework, we use the terms *always* and *some-time*. These modalities were studied first by the Megarians³, then by Aristotle and the Stoic. Despite their variant, these modalities are linked to the universal \forall and particular \exists quantifiers.

Manna and Pnuelli [23,24], defined three important classes of temporal properties of concurrent programs that are investigated inside the modal

³A school founded by Euclid of Megaric, student of Socrates, like Plato, this school was concurrent of Aristotle's one.

temporal framework. They are invariance, liveness and precedence properties. The first two ones are closely related to the two basis modalities \Box , which is interpreted as *always from now on* and \Diamond which is interpreted as *sometimes* or *at least one time from now on*. The third one is an extension of the \Diamond class, proposed by Manna and Pnuelli, closed to the \mathcal{U} , the *until* modality in order to capture precedence in general. These three classes are available for temporal properties in general. We first review these three classes and then we revisit their example inside our formal languages framework. Our purpose is not to criticize their solution, but only to translate it inside our model.

6.1. Temporal properties types

The first class is the class of invariance properties. These are properties that can be expressed by a temporal formula of the form: $\Box\psi$ or $\varphi \Rightarrow \Box\psi$. Such a formula, stated for a program P , says that every computation of P continuously satisfies ψ throughout the rest of the computation either from the beginning (first formula) or whenever φ becomes true. Among properties falling into this class are: partial correctness, error-free behavior, mutual exclusion and absence of deadlocks.

The second set, associated to the *sometimes* modality defines the *liveness* properties class. These properties are expressible by temporal formulas of the form: $\Diamond\psi$ or $\varphi \Rightarrow \Diamond\psi$. In both cases these formulas guarantee the occurrence of some event ψ ; in the first case unconditionally and in the second case conditional on an earlier occurrence of the event φ . Among properties falling into this class are: total correctness, termination, accessibility, lack of individual starvation, and responsiveness.

The third class is the precedence properties class which is very well-known inside the artificial intelligence community. In a broad sense, Manna and Pnuelli asserted that precedence properties are all the properties that are expressible using the *until* operator \mathcal{U} in formulas such as $\chi\mathcal{U}\psi$ or $\varphi \Rightarrow \chi\mathcal{U}\psi$. In both cases the formulas again guarantee the occurrence of the event ψ , but they also ensure that from now until that occurrence, χ will continuously hold. Among properties falling into the *until* class are strict (FIFO) responsiveness, and bounded overtaking. The meaning they give to the precedence operator, that is to the formula $p < q$,

is that q eventually happens, and $p < q$ is automatically satisfied if q never happens. Hence we have:

$$p < q \equiv \neg((\neg p)\mathcal{U}q)$$

We are now ready to embark inside the problem of allocating a single resource between several requesters as explained in [24]. In this paper we recall almost every thing about their specification, because this kind of problem is of general interest for all complex systems.

6.2. The Allocation Problem

Let us consider a program G (granter) serving as an allocator of a single resource between several processes (requesters) R_1, \dots, R_k competing for the resource. Let each R_i communicate with G by means of two boolean variables: r_i and g_i . The variable r_i is set to *true* ($= 1$) by the requester to signal a request for the resource. Once R_i has the resource it signals its release by setting it to *false* ($= 0$). The allocator G signals R_i that the resource is granted to him by setting g_i to *true*. Having obtained a release signal from R_i , which is indicated by $r_i = \text{false}$, some time later, it will appropriate the resource by setting g_i to *false*.

6.2.1. Properties of the system described inside the modal logic framework.

Several obvious and important properties of this system belong to the invariance and liveness classes.

An invariant property. Insuring that the resource is granted to at most one requester at a time is an invariant property:

$$\Box\left(\sum_{i=1}^k g_i\right) \leq 1$$

A liveness property: responsiveness The important property that ensures responsiveness, i.e. which guarantees that every request r_i will eventually be granted by setting g_i to *true* is a liveness property:

$$(\forall i)(1 \leq i \leq k)(r_i \Rightarrow \Diamond g_i)$$

Precedence properties. Two precedence properties are set.

– *An absolute precedence property: Absence of Unsolicited Response.* An important but often overlooked desired feature is that the resource will not be granted to a party who has not requested it. A similar property in the context of a communication network is that every message received must have been sent by somebody. This is expressible by the temporal formula:

$$\neg g_i \Rightarrow (r_i < g_i)$$

The formula states that if presently g_i is *false*, i.e., R_i does not presently have the resource, then before the resource will be granted to R_i the next time, R_i must signal a request by setting r_i to *true*.
 – *A relative precedence property: a Strict (FIFO) Responsiveness.* Sometimes the weak commitment of eventually responding to request is not sufficient. At the other extreme we may insist that responses are ordered in a sequence paralleling the order of arrival of the corresponding requests. Thus if requester R_i succeeded in placing in request before requester R_j , the grant to R_i should precede the grant to R_j . A straightforward translation of this sentence yields the following intuitive but slightly imprecise expression: $(r_i < r_j) \Rightarrow (g_i < g_j)$. A more precise expression is $(\forall i, j)(i \neq j)(1 \leq i, j \leq k)$

$$((r_i \wedge \neg r_j \wedge \neg g_j) \Rightarrow (\neg g_j \mathcal{U} g_i)).$$

It states that if ever we find ourselves in a situation where r_i is presently on, r_j and g_j are both off, then we are guaranteed to eventually get a g_i , and until that moment, no grant will be made to R_j . Note that $r_i \wedge \neg r_j$ implies that R_i 's request precedes R_j 's request, which has not been materialized yet.

We implicitly rely here on the assumption that once a request has been made, it is not withdrawn until the request has been honored. This assumption can also be made explicit as part of the specification, using another precedence expression:

$$r_i \Rightarrow g_i < (\neg r_i).$$

Note that while all the earlier properties are requirements from the granter, and should be viewed as the *post-condition* part of the specification, this requirement is the responsibility of the requesters. It can be viewed as part of the *pre-condition* of the specification.

Two assumptions are also implicitly used but not mentioned:

1. a process is not allowed to make an other request until he has given the resource back,
2. there are as many requests from R_i as grants for R_i .

We now are leaving the way Manna and Pnuelli have resolved the problem inside the modal logic framework: finding an abstract computation model based on sequences of transitions and states, a proof system [24].

6.3. revisitation of the allocation problem inside the S-languages framework

6.3.1. Objects and relations representations

Our formalization attempts to translate any information into a temporal information. The two boolean variables introduced in the formulation of the problem do not belong to the problem but to one of its data interpretation. These boolean variables are evolving through the timeline. It is natural to represent the values of boolean variables evolving through the time as characteristics function of boolean variables on a linear order that can be called temporal boolean functions [7,15,16]. Inside a determined and bounded period of time, the temporal boolean functions r_i (resp. g_i) can be interpreted as a chain of intervals with n_i intervals, where n_i is the number of requests (resp. grants). This number is exactly known at the end of the fixed period of time. Any interval is a maximal period inside which the value of r_i (resp. g_i) is *true*.

A priori, for the general specification, either we can choose to write n_i as an indeterminate number or to set $*$ for saying that there is a finite but unknown number. To each requesters, we provide $2k$ chains of intervals written on their identities alphabet $X = \{r_1, g_1, \dots, r_k, g_k, \}$. Any language L that satisfies the problem is such that its Parikh number is $\tilde{L} \subseteq \underbrace{((2, 2)\mathbb{N}, \dots, (2, 2)\mathbb{N})}_{k \text{ times}}$. But for the sake

of an easier reading, we will use the following alphabet $X = \{r_1, \bar{r}_1, g_1, \bar{g}_1, \dots, r_k, \bar{r}_k, g_k, \bar{g}_k\}$, which allows to make a distinction between the beginning of the interval (no marked letter) and the end of the interval (marked letter) so that any language L that satisfies the problem is such that its Parikh number is $\tilde{L} \subseteq \underbrace{((1, 1, 1, 1)\mathbb{N}, \dots, (1, 1, 1, 1)\mathbb{N})}_{k \text{ times}}$.

6.3.2. The specification.

The invariant property. Insuring that the resource is granted to at most one requester at a time can be interpreted like this: this constraint only concerns granters. After having read a letter g_i – a beginning of an allocation to R_i –, the first following occurrence of a letter of type g or \bar{g} inside the S-word is necessarily a \bar{g}_i letter (the end of the current allocation to R_i). This constraint is formulated by the S-language

$$L_1 = \int_X (g_1 \bar{g}_1, \dots, g_k \bar{g}_k)^*.$$

Hence we have $L \subseteq L_1$.

The liveness property. The guaranty that every request r_i will eventually be granted concerns each R_i individually. This organizes every couple of S-words $(r_i \bar{r}_i)^*$ and $(g_i \bar{g}_i)^*$ saying that after each occurrence of a letter r_i , the first occurrence of a letter among the set $\{r_i, \bar{r}_i, g_i, \bar{g}_i\}$ is an occurrence of the letter g_i . There is no constraint between the occurrences of the letters \bar{r}_i and \bar{g}_i that is to say that we get

$$L \subseteq \bigcap_{1 \leq i \leq k} \int_X (r_i g_i [\bar{r}_i, \bar{g}_i])^*$$

But we can give a more precise S-language, because we know that the sequence of actions related to any request is the following: request, allocation, release and deallocation. So more precisely we can give

$$L \subseteq L_2 = \bigcap_{1 \leq i \leq k} \int_X (r_i g_i \bar{r}_i \bar{g}_i)^*.$$

The two precedence properties.

– *Absence of Unsolicited Response.* A resource will be not granted to a party who has not requested it. This constraint is already written in the preceding constraint.

– *Strict (FIFO) Responsiveness.* Grants are ordered in the same order as the corresponding requests. This concerns how the occurrences of r_i , r_j , g_i , g_j are shuffled together. The case where $r_i = r_j$ has been already taken into account in L_2 . Restricted to the alphabet $\{r_i, r_j, g_i, g_j\}$, if R_i requests before R_j , either R_i is granted before the request of R_j or after⁴. But R_j can request before

R_i . That is we get four cases that can be organized as product but not as a shuffle. Hence we have

$$L \subseteq L_3 = \bigcap_{1 \leq i < j \leq k} \int_X (r_i g_i, r_j g_j, r_i r_j g_i g_j, r_j r_i g_j g_i)^*$$

6.3.3. The solution.

All constraints are now to be specified in terms of S-languages. Every S-word contained in $L_1 \cap L_2 \cap L_3$ satisfies the problem, hence the solution is $L = \int_X (g_1 \bar{g}_1, \dots, g_k \bar{g}_k)^* \cap \bigcap_{1 \leq i \leq k} \int_X (r_i g_i \bar{r}_i \bar{g}_i)^* \cap \bigcap_{1 \leq i \neq j \leq k} \int_X (r_i g_i, r_j g_j, r_i r_j g_i g_j, r_j r_i g_j g_i)^*$

If the system can't realize two tasks simultaneously, the solution will be $L \cap X^*$.

6.3.4. Example

Let us suppose that there are three requesters and the order of the requests is $R_1 R_2 R_3 R_1 R_3$. The temporal objects are the four chains expressed with the S-words – that are also words – $r_1 \bar{r}_1 r_1 \bar{r}_1$, $g_1 \bar{g}_1 g_1 \bar{g}_1$, $r_3 \bar{r}_3 r_3 \bar{r}_3$, $g_3 \bar{g}_3 g_3 \bar{g}_3$, and the intervals expressed with the S-words $r_2 \bar{r}_2$, $g_2 \bar{g}_2$. The set of all possible situations is given by the language $[r_1 \bar{r}_1 r_1 \bar{r}_1 | g_1 \bar{g}_1 g_1 \bar{g}_1 | r_3 \bar{r}_3 r_3 \bar{r}_3 | g_3 \bar{g}_3 g_3 \bar{g}_3 | r_2 \bar{r}_2 | g_2 \bar{g}_2]$. That is the S-language with Parikh vector $((2, 2), (2, 2), (1, 1), (1, 1), (2, 2), (2, 2))$ on the ordered alphabet $\{r_1, \bar{r}_1, g_1, \bar{g}_1, r_2, \bar{r}_2, g_2, \bar{g}_2, r_3, \bar{r}_3, g_3, \bar{g}_3\}$.

The requests order $R_1 R_2 R_3 R_1 R_3$ induces the following sequence between the letters of type r :

$$r_1 r_2 r_3 r_1 r_3.$$

$$L_2 = \int_X \{r_1 g_1 \bar{r}_1 \bar{g}_1 r_1 g_1 \bar{r}_1 \bar{g}_1, r_2 g_2 \bar{r}_2 \bar{g}_2, r_3 g_3 \bar{r}_3 \bar{g}_3 - r_3 g_3 \bar{r}_3 \bar{g}_3\}.$$

The condition on the r letters and the expression of sL_3 allows to substitute to L_3 , the language L'_3 :

$$L'_3 = \int_X g_1 g_2 g_3 g_1 g_3.$$

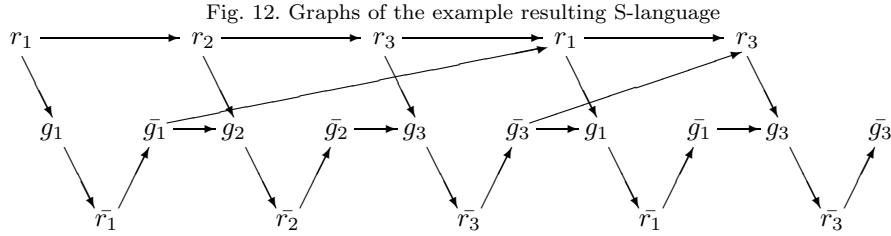
$$L_1 = \int_X g_1 \bar{g}_1 g_2 \bar{g}_2 g_3 \bar{g}_3 g_1 \bar{g}_1 g_3 \bar{g}_3.$$

The shuffle of all these fragments are depicted by a Hasse graph of the precedence ordering on the instants corresponding to the bounds of intervals, that is the letters in figure 12. This is a graphical representation of the resulting S-language L .

7. Conclusion

In this paper, we have presented the S-languages framework and shown how to represent and to reason on qualitative temporal problem. The Hasse Diagram we provide for the allocation problem has to be compared with the temporally labeled graph of Gerevini and Schubert [12].

⁴The system is not allowed to receive simultaneous requests.



Different implementations have been made in order to improve the complexity on the computations. They take benefits of the algorithms used for computing operations in formal languages theory, and the use of automata theory. The problems come, as usual, from the parts of S-languages that have to be broken into disjoint parts, in order to go on in the computation.

Two implementations has already been made, concerning the interval algebra. The first one is based on the notion of pattern [11]. The second one [27] applies it on the linguistic model of Desclés based on topological intervals [10]. Bounds of intervals are labelled in order to mention whether an interval is open or closed. As $\mathcal{L}(p_1, \dots, p_n)$ is a lattice, we work on convex parts, following the approach of Freksa [14]. These prototypes suggest that it may be better to compute in two steps: first accepting all situations, even those not allowed situations, and second without forbidden situations, rather than to compute directly the good solution.

Acknowledgements

I would thank Jean-Michel Autebert and Jérôme Cardot for their fruitful comments on the text.

References

- [1] James F. Allen, An Interval-Based Representation of Temporal Knowledge. Proc. 7th, Vancouver, Canada, August 1981, p221-226
- [2] James F. Allen. Maintening Knowledge about Temporal Interval. *Comm. ACM* 26-11, (1983) 832-843.
- [3] Jean-Michel Autebert, *Langages Algébriques*. Masson, Paris, 1987.
- [4] Jean-Michel Autebert, Matthieu Latapy, and Sylviane R. Schwer, *Le treillis des chemins de Delannoy*, Discrete Math., 258 (2002) pp. 225-234.
- [5] Jean-Michel Autebert, Sylviane R. Schwer, On generalized Delannoy Paths, *SIAM Journal on Discrete Mathematics* 16 (2) (2003) 208-223.
- [6] Hélène Bestougeff and Gérard Ligozat, *Outils logiques pour le traitement du temps* Masson (1989).
- [7] Maroua Bouzid, Antoni Ligeza, Temporal Logic Based on Characteristic Function. *19th Annual German Conference on Artificial Intelligence*, Lecture Notes in Artificial Intelligence, 981(1995) 221-232.
- [8] Roy H. Campbell and Nico Haberman, The Specification of Process Synchronization by Paths Expressions. *Lecture Note in Computer Science* Springer-Verlag 16 (1974) 89-102.
- [9] Henri Delannoy, *Emploi de l'échiquier pour la résolution de certains problèmes de probabilités*, Comptes-Rendus du Congrès annuel de l'Association Française pour l'Avancement des Sciences, vol 24, p.70-90, Bordeaux, 1895.
- [10] Jean-Pierres Desclés. State, Event, Process, and topologie. *General Linguistics*, 29 (3), (1990) 159-200
- [11] Michel Dubois, Sylviane R. Schwer. Classification topologique des ensembles convexes de Allen, *proceeding of the 12th Congrès Francophone AFRIF-AFIA. R.F.I.A. 2000*, Paris, '2-4 février 2000) III 59 - 68.
- [12] A. Gerevini, L. Schubert. Efficient algorithms for qualitative reasoning about time. *Artificial Intelligence*, 74 (1995) 207-248.
- [13] Seymour Ginsburg. *The Mathematical Theory of Context-Free Languages*. McGraw-Hill, 1966.
- [14] Christian Freksa, Temporal reasoning based on semi-intervals, *Artificial Intelligence*, 54 (1991) 199-227.
- [15] Setthachaï Jungjariyanonn, Sylviane R. Schwer, Condition as Temporal Functions. *Proceedings of the third Basque International Workshop on Information Technology*. (Biarritz, 2-4 juillet 1997), 148-156.
- [16] Setthachaï Jungjariyanonn, Sylviane R. Schwer, Extended Boolean Computations. *Proceedings of Workshop 8 (Spatial and Temporal Reasoning) of the 15th European Conference on Artificial Intelligence* (Lyon, France-23 Juillet 2002) 63-68
- [17] Hans Kamp, Events, Instants and Temporal Reference, in *Semantics from Different points of view*, Bauerte, R., Egli, U., von Stechow A., (eds), Spronger Verlag, p. 376-417, 1979.
- [18] E. Yu Kandrashina, Representation of Temporal Knowledge. Proceedings of the 8th International Joint Conference on Artificial Intelligence, 8-12 August 1983, Karlsruhe, West Germany.
- [19] al-Khatib Lina, Reasoning with non-convex time intervals. Thesis Melbourne, Florida, september 1994.

- [20] Peter Ladkin. Time Representation: A Taxonomy of Interval Relations. AAAI pages 360 – 366, (1986).
- [21] Peter B. Ladkin and Roger D. Maddux, On Binary Constraint Problems, in Journal of the ACM 41(3):435-469, May 1994. This paper is a substantial reworking of the technical report: On Binary Constraint Networks, by Peter B. Ladkin and Roger Maddux, Technical Report KES.U.88.8, Kestrel Institute, 1988.
- [22] Gérard Ligozat. On generalized interval calculi, *Proceedings of the AAAI, Anaheim, C.A.*, (1991) 234–240
- [23] Zohar Manna, Amir Pnueli ; Verification of Concurrent Programs: The temporal proof principles. *Automata, Languages and Programming*, Lecture Notes in Computer Science 131 (Springer, Berlin 1983) 200–252.
- [24] Zohar Manna, Amir Pnueli ; Proving precedence properties: The temporal way. *Automata, Languages and Programming*, Lecture Notes in Computer Science 154 (Springer, Berlin 1983) 491–512.
- [25] Klaus Nökel, Convex Relations Between Time Intervals. *Proceedings of the AAAI, Boston, MA.* (1990) 721–727
- [26] Bernard A. Nudel Consistent-labelling Problems and their Algorithms, *Artificial Intelligence* 21 (1983) 135–178
- [27] Etienne Picard Etude des structures de données et des algorithmes pour une implantation des éléments et relations temporelles en vue du traitement automatique de la temporalité dans le langage naturel. *Mémoire de DEA. Université Paris Sorbonne* (Paris IV), septembre 2003.
- [28] David A. Randell, Anthony. C. Cohn, Zheng Cui ; Computing Transitivity Tables: A Challenge For Automated Theorem Provers. *Proc. CADE, LNCS, Springer Verlag*, (1992)
- [29] Bertrand Russell, *Our Knowledge of the external World*, G. Allen & Unwin, London, 1914, réédition 1924
- [30] Sylviane R. Schwer. Raisonnement Temporel : les mots pour le dire. *Rapport interne LIPN* , 1997
- [31] Sylviane R. Schwer, S-arrangements avec répétitions, *Comptes Rendus de l'Académie des Sciences de Paris*, Série I 334 (2002) 261–266.
- [32] Neil J.A. Sloane, *sequence A001850/M2942*, An On-Line Version of the Encyclopedia of Integer Sequences, <http://www.research.att.com/njas/sequences/eisonline.html>
- [33] Marc Vilain, A system for reasoning about time. *Proceedings of the AAAI* (1982) 197–201.
- [34] Marc Vilain, Henry Kautz, Constraint Propagation Algorithms for Temporal Reasoning, *Proceedings of the AAAI* (1986) 377-382
- [35] Eric Weisstein, *CRC Concise Encyclopedia of Mathematics*, CRC Press, 1999.